ECE503 Spring 2014 Project 1

This project is worth 100 points
and is due by 6:00pm Monday 17-March-2013

In this project, you will use MATLAB to implement different quantization schemes and to experimentally verify the signal to noise ratio of standard and oversampled uniform quantizers with and without noise shaping.

1 Standard Uniform Quantization

In this part of the project you will implement a standard uniform quantizer as shown in Oppenheim and Shafer Fig. 4.53 and simulate it with random inputs. The quantizer is specified by the scale parameter X_m and the number of bits B with step size $\Delta = \frac{X_m}{2^B}$ (see Fig. 4.54 in Oppenheim and Shafer). You can implement the quantizer as your own custom function or use the MATLAB function `quantiz` which requires you to specify codebook and partition vectors. Be sure to carefully read the documentation on this function if you decide to use it. A good first step is to make sure you can correctly specify the codebook and partition vectors given X_m and B (try the $B = 3$ case first and compare your results to Fig. 4.54 in Oppenheim and Shafer).

Note that the assignment is to implement an actual quantizer, not the additive quantization noise model used for analysis. You should compute the quantization error by computing the difference between the unquantized inputs and the quantized outputs of your quantizer.

To confirm that your quantizer is working correctly, you should test your quantizer by generating a random input, quantizing it, and computing the variance of the quantization error for different values of $B \in \{2, \ldots, 16\}$. You should then be able to generate a plot of the experimental quantizer SNR versus the theoretical quantizer SNR

$$\text{SNR}_Q = 6.02B + 10.8 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right)$$

as a function of B for a given X_m and σ_x. Some suggestions:

1. Recall that our analysis leading to the theoretical quantizer SNR assumes no quantizer saturation. Quantizer saturation will cause your experimental results to be much worse than the theoretical predictions. Even if quantizer saturation occurs rarely, it may affect your results for large values of B. To ensure no quantizer saturation, you can either clip Gaussian-distributed inputs prior to quantization or just generate uniformly-distributed inputs on some range $[-a, a]$ that is less than the full-scale range of the quantizer. In either case, your quantizer inputs should use most of the full-scale range of the quantizer.

2. Make sure your plots have axis labels and legends as necessary. Part of your grade is based on the clarity of your results.

3. Your experimental results and theoretical results should line up quite nicely if you quantize at least a few thousand samples for each value of B.

2 Oversampled Uniform Quantization

In this part of the project you will implement an oversampled uniform quantizer as shown in Oppenheim and Shafer Fig. 4.65 and simulate it with random inputs. You should get part 1 of the project working correctly first since that will form the basis for this part of the project. As in part 1, you should implement an actual quantizer and not the additive quantization noise model (don't implement Oppenheim and Shafer Fig. 4.66).

To confirm that your oversampled quantizer is working correctly, you should test your quantizer by generating a random input, quantizing it, and computing the variance of the quantization error for different values of $B \in \{2, \ldots, 16\}$. You should then be able to generate a plot of the experimental quantizer SNR versus the theoretical quantizer SNR

$$\text{SNR}_Q = 6.02B + 10.8 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right) + 10\log_{10}(M).$$

The basic steps are as follows:

1. Generate a random unquantized signal that spans most of the full-scale range of the quantizer.
2. Upsample (and interpolate) the random signal.
3. Quantize the upsampled signal. You should be able to re-use your part 1 code here.
4. Lowpass filter and downsample.
5. Compute the quantization error and its variance.

Since the filtering operations in Matlab are not ideal, it can be trickier to get this case working correctly than part 1. Here are some suggestions for your implementation:

1. Apply a lowpass filter to this random input signal before upsampling. The parameters of the LPF don't matter very much; you can set the cutoff frequency to $\pi/2$ or any reasonable value. The idea is to attenuate the high frequency components of the input signal to give the upsample/downsample operations some filtering margin. You can use `fdatool` to easily generate a suitable LPF and the `filter` command to perform the filtering. This filtered signal should be considered the input to the upsampling quantizer when computing the quantization error in step 5.
2. Use the MATLAB command `resample` rather than `interpolate` and `decimate` and pay attention to the parameter that allows you to adjust the length of the interpolation/decimation filter. It defaults to 10, but you may want to try other values to see if you can get better results.
3. Since the filtering operations are not ideal, there may be large quantization errors at the beginning and/or end of your simulation. These transient errors occur because we are only processing a finite number of samples. You should discard these transients before computing the variance of the quantization error.
4. Test your oversampled quantizer for at least a few values of M.

3 Oversampled Uniform Quantization with Noise Shaping

In this part of the project you will extend your results from part 2 to implement an oversampled uniform quantizer with noise shaping as shown in Oppenheim and Shafer Fig. 4.70b and simulate it with random inputs. You should get part 2 of the project working correctly first since that will form the basis for this part of the project. As in part 2, you should implement an actual quantizer and not the additive quantization noise model (don't implement Oppenheim and Shafer Fig. 4.71). You should test your quantizer as in the previous parts and compare to the analytical performance prediction

$$\mathrm{SNR}_Q = 6.02B + 5.62 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right) + 30\log_{10}(M).$$

All of the suggested procedures for part 2 apply to this part.

Note that this quantizer will probably run much slower than parts 1 and 2 (since you will have to loop through the input signal on a sample-by-sample basis), so you should test your oversampled quantizer with noise shaping for $B \in \{2, \ldots, 8\}$ first to make sure it is working correctly. When you are confident it is working correctly, then you should test your quantizer to larger values of B. Also test your oversampled quantizer with noise shaping for at least a few values of M.

4 Project Submission

Submit the following to the TA and instructor via email prior to the due date/time:

- Your final Matlab code (a separate .m file for each part). Each file should be well commented, properly indented, and should have your name in the header. If you wrote any custom functions, they should also be included. The TA/instructor should be able to run your code directly without errors. Part of your grade will be based on the clarity of your code.

- A report in .pdf format, no longer than three single-sided pages, explaining your code and the main results. Your font size must not be smaller than 10 point. Provide plot(s) showing the theoretical and experimental quantizer performance as a function of B and M for given X_m and σ_x. Explain your methodology and summarize the main results. Also be sure to discuss any deviation between your experimental results and the analysis. You are encouraged to include block diagrams and other plots as necessary to explain your work.